AD-A055 868 SRI INTERNATIONAL MENLO PARK CA F/G 9/2
DECIDING LINEAR INEQUALITIES BY COMPUTING LOOP RESIDUES.(U)
MAR 78 R SHOSTAK F44620-73-C-0068
UNCLASSIFIED AFOSR-TR-78-1095 NL

1 OF 1
AD A055 868

END
DATE FILMED
8-78
DDC

DECIDING LINEAR INEQUALITIES BY COMPUTING LOOP RESIDUES

Interim rept.,

By: Robert Shostak
SRI International
333 Ravenswood Ave.
Menlo Park, California 94025

10 March 1978

19 p.

2394

A2

This work was supported in part by the National Science Foundation under Grant MCS76-81425, Air Force Office of Scientific Research under Contract F44620-73-C-0068, and Rome Air Development Center under Contract F30602-78-C-0031.

410 281

SRI International
333 Ravenswood Ave. • Menlo Park, California 94025
(415) 326-6200 • Cable: STANRES, Menlo Park • TWX: 910-373-1246

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DDC
This technical report has been reviewed and is
approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 78-1095	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) DECIDING LINEAR INEQUALITIES BY COMPUTING LOOP RESIDUES		5. TYPE OF REPORT & PERIOD COVERED Interim 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Robert Shostak		8. CONTRACT OR GRANT NUMBER(s) F44620-73-C-0068
9. PERFORMING ORGANIZATION NAME AND ADDRESS SRI International 333 Ravenswood Avenue Menlo Park, California 94025		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332		12. REPORT DATE March 10, 1978 13. NUMBER OF PAGES 18
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

< or =

The method is generalized

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

V. Pratt [18] has shown that the real and integer feasibility of sets of linear inequalities of the form $x \leq y + c$ can be decided quickly by examining the loops in certain graphs. ~~We generalize Pratt's method~~ first to real feasibility of inequalities in two variables and arbitrary coefficients, and ultimately to real feasibility of arbitrary sets of linear inequalities. The method is well suited to applications in program verification.

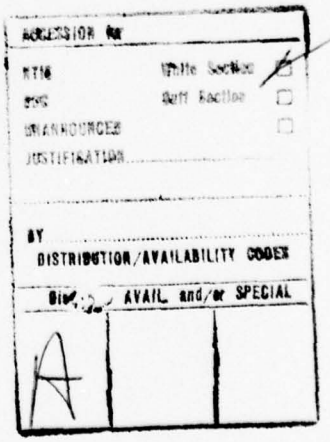
DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT

V. Pratt [18] has shown that the real and integer feasibility of sets of linear inequalities of the form $x \leqslant y + c$ can be decided quickly by examining the loops in certain graphs. We generalize Pratt's method, first to real feasibility of inequalities in two variables and arbitrary coefficients, and ultimately to real feasibility of arbitrary sets of linear inequalities. The method is well suited to applications in program verification.



1. Introduction

Procedures for deciding whether a given set of linear inequalities has solutions often play an important role in deductive systems for program verification. Array bounds checks and tests on index variables are but two of the many common programming constructs that give rise to formulas involving inequalities. A number of approaches have been used to decide the feasibility of sets of inequalities [2,7,8,14,21], ranging from goal-driven rewriting mechanisms [25] to the powerful simplex techniques [7] of linear programming. Some simple methods are well suited to the small, trivial problems that most often arise, but are insufficiently general. Full-scale simplex techniques, on the other hand, are general and fast for medium to large problems, but do not take advantage of the trivial structure of the small problems encountered most frequently.

The algorithm presented here retains the generality needed in the exceptional case, without sacrifice of speed and simplicity in the more typical situation. It builds on V. Pratt's observation [18,16] that most of the inequalities that arise from verification conditions are of the form $x \leq y + c$, where x and y are variables and c is a constant. Pratt showed that a conjunction of such inequalities can be decided quickly by examining the loops of a graph constructed from the inequalities of the conjunction. We generalize this approach, first to inequalities with no more than two variables and with arbitrary coefficients, and then to arbitrary linear inequalities. Our generalization reduces to Pratt's test for inputs having the simple structure he describes.

The discussion is presented in six sections. Sections 2 and 3 are concerned with preliminary definitions and with a statement of the method for inequalities with two variables and arbitrary coefficients. Section 4 discusses issues of complexity and usefulness for integer problems, and relates the method to Pratt's. Sections 5 and 6 deal with the extension of the method to sets having strict inequalities and to sets with arbitrary linear inequalities. The last section presents a proof of the theorem that underlies the method.

2. Definitions

Let S be a set of linear inequalities each of whose members can be written in the form $ax + by \leqslant c$, where x, y are real variables and a, b, c are reals. Without loss of generality, we require that all variables appearing in S other than a special variable v_0, called the *zero variable*, have nonzero coefficients. We also assume that v_0 appears only with coefficient zero.

Construct an undirected multi-graph G from S as follows. Give G a vertex for each variable occurring in S and an edge for each inequality. Let the edge associated with an inequality $ax + by \leqslant c$ connect the vertex for x with the vertex for y. Label each vertex with its associated variable* and each edge with its associated inequality. G is said to be the *graph for S*.

Now let P be a path through G, given by a sequence $v_1, v_2, \ldots, v_{n+1}$ of vertices and a sequence $e_1, e_2, \ldots, e_n$ of edges, $n \geqslant 1$. The *triple sequence* for P is given by:

$$\langle a_1, b_1, c_1\rangle, \langle a_2, b_2, c_2\rangle, \ldots, \langle a_n, b_n, c_n\rangle \quad ,$$

where for each i, $1 \leqslant i \leqslant n$, $a_i v_i + b_i v_{i+1} \leqslant c_i$ *is the inequality labeling* e_i.** P is *admissible* if, for $1 \leqslant i \leqslant n - 1$, b_i and a_{i+1} have opposite signs; i.e., one is strictly positive and the other is negative.

Intuitively, admissible paths correspond to sequences of inequalities that form transitivity chains. For example, the sequence $x \leqslant y$, $y \leqslant z$, $z \leqslant 3$ gives rise to an admissible path, as does

$$2x \geqslant 3y - 4,\ 2y \geqslant 4 - z,\ -z \geqslant x \quad .$$

Note that the sequence:

$$x \leqslant y,\ y \leqslant z,\ -z \leqslant r$$

---

*In what follows, it is notationally convenient to write v for both the variable v and the vertex associated with that variable.

**In the case where v_i and v_{i+1} happen to be identical (i.e., e_i is a self-loop), an arbitrary choice is made as to the ordering of the first two components of the associated triple.

cannot label an admissible path, since the coefficients of z have the wrong relative signs.

A path is a *loop* if its first and last vertices are identical. A loop is *simple* if its intermediate vertices are distinct.

Note that the reverse of an admissible loop is always admissible, and that the cyclic permutations of a loop P are admissible if and only if a_1 and b_n are of opposite sign, where $\langle a_1, b_1, c_1\rangle \ldots \langle a_n, b_n, c_n\rangle$ is the triple sequence for P. In this case, we say P is *permutable*. Note also that, since v_0 appears in S only with coefficient 0, no admissible loop with initial vertex v_0 is permutable.

Now define, for a given admissible path P, the *residue inequality of P* as the inequality obtained from P by applying transitivity to the inequalities labeling its edges. For example, if the inequalities along P are

$$x \leqslant 2y + 1,\ y \leqslant 2 - 3z,\ -z \leqslant w \quad ,$$

we have:

$$x \leqslant 2y + 1 \leqslant 2(2 - 3z) + 1 \leqslant 2(2 + 3w) + 1 = 6w + 5$$

The residue inequality of P is thus $x - 6w \leqslant 5$.

More formally, define the *residue* r_p of P as the triple $\langle a_p, b_p, c_p\rangle$ given by:

$$\langle a_p, b_p, c_p\rangle = \langle a_1, b_1, c_1\rangle * \langle a_2, b_2, c_2\rangle * \ldots * \langle a_n, b_n, c_n\rangle \quad ,$$

where $\langle a_1, b_1, c_1\rangle \ldots \langle a_n, b_n, c_n\rangle$ is the triple sequence for P and where * is the binary operation on triples defined by:

$$\langle a, b, c\rangle * \langle a', b', c'\rangle = \langle kaa', -kbb', k(ca' - c'b)\rangle$$

$$\text{and } k = \frac{a'}{|a'|} \quad .$$

The *residue inequality* of P is then given by $a_p x + b_p y \leqslant c_p$, where x and y are the first and last vertices, respectively, of P.

It is straightforward to show that * is associative, so that r_p is in fact uniquely defined. The idea that the residue inequality of a path is implied by the inequalities labeling the path is expressed in the following lemma:

Lemma 1. Any point (i.e., assignment of reals to variables) that satisfies the inequalities labeling an admissible path P also satisfies the residue inequality of P.

Pf. Straightforward by induction on the length of P.

3. Procedure for Inequalities with Two Variables

In the case where P is a loop with initial vertex, say, x, Lemma 1 asserts that any point satisfying the inequalities along P must also satisfy $a_p x + b_p x \leq c_p$. If it happens that $a_p + b_p = 0$ and $c_p < 0$, the residue inequality of P is false, and we say that P is an *infeasible loop.*

It follows that a set S of inequalities is unsatisfiable if the graph G for S has an infeasible loop. The converse, however, does not hold in general. Figure 1, for example, shows the graph for $S = \{x \leq y,\ 2x + y \leq 1,\ z \leq x,\ w \leq z,\ z \leq 1 + w,\ z \geq \tfrac{1}{2}\}$. Although S is unsatisfiable, the graph has no infeasible loops, simple or otherwise.

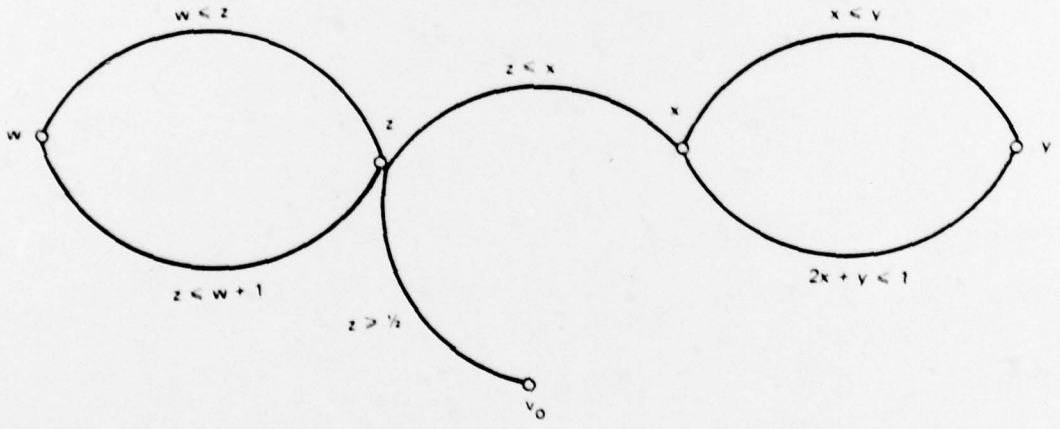


FIGURE 1 GRAPH G FOR $S = \{x \leq y,\ 2x + y \leq 1,\ z \leq x,\ w \leq z,\ z \leq w + 1,\ z \geq \tfrac{1}{2}\}$

The gist of our main theorem is that G can be modified to obtain a graph G′ that has an infeasible simple loop if and only if S is unsatisfiable:

Definition: Let G be the graph for S. Obtain a *closure* G′ of G by adding, for each simple admissible loop P (modulo permutation and reversal) of G a new edge labelled with the residue inequality of P.

Note that closures are not necessarily unique, since the initial vertex of each permutable loop can be chosen arbitrarily.

Theorem: S is unsatisfiable if and only if G′ has an infeasible simple loop.

Figure 2 shows the unique closure of the graph of Figure 1. Note that the only loop of G contributing an edge to G′ is the xyx loop. The v_0xzv_0 loop of G′ is infeasible (having residue $\langle 0, 0, -1/3 \rangle$); hence the example S, according to the theorem, must be unsatisfiable.

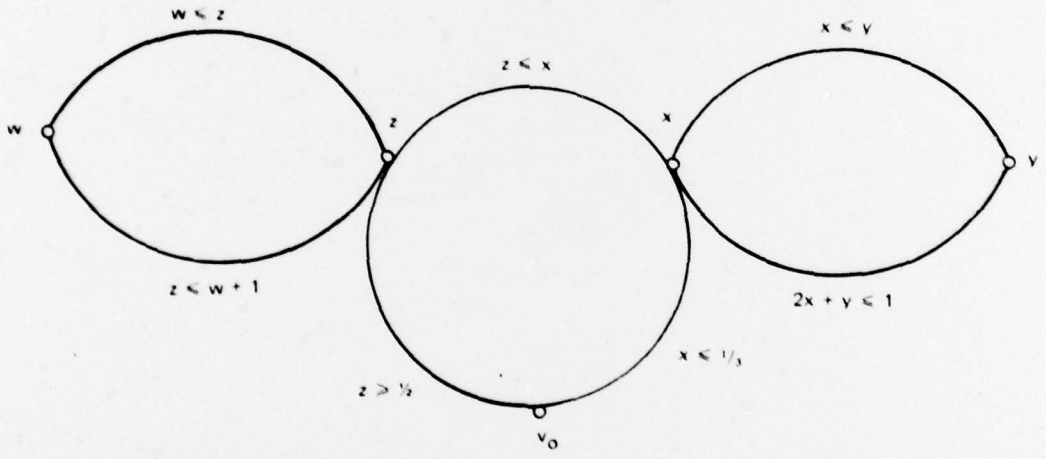


FIGURE 2 CLOSURE OF G

We show later that any cyclic permutation of an infeasible permutable loop is itself infeasible, and that the reverse of an infeasible loop is also infeasible. We thus have the following decision procedure for satisfiability of S:

(1) The simple admissible loops of G are enumerated modulo cyclic permutation and reversal, and their residues are computed. If any loops are found to be infeasible, S is unsatisfiable.

(2) Otherwise, the closure of G is formed by adding a new edge for each residue inequality. The residues of all newly formed simple admissible loops are now computed. If any are found to be infeasible, S is unsatisfiable. Otherwise S has solutions.

Note that this procedure, as stated, does not actually construct a solution if S is feasible. The proof of the main theorem, given in Section 7, provides such a construction. Note also that the new admissible loops formed in (2) must have initial vertex v_0.

4. Efficiency and Other Issues

Any implementation of the procedure must, of course, incorporate some means of generating the simple loops of a graph. For this purpose, several algorithms exist (Johnson [13], Read and Tarjan [19], Szwarcfiter and Lauer [23]) that operate in time order $\mathcal{O}(|V| + |E|)$, and space order $|V| + |E|$, where ℓ is the number of loops generated. These algorithms are easily modified to generate only admissible loops without adversely affecting efficiency. Since each loop has length on the order of $|V|$, these algorithms require little more time than that needed for output. A graph may, of course, have quite a few simple loops – exponentially many (in $|E|$), in fact, in the worst case. One can show that the procedure we have described, like the simplex method, exhibits exponential worst-case asymptotic behavior.

In practice, however, one does not encounter such behavior. The sets of inequalities that arise from verification conditions usually have the form of transitivity chains. The corresponding graphs are treelike, seldom having more than a few loops. Most of the loops that do occur are 2-loops, which are easily tested at the time the graph is constructed.

V. Pratt [18] has noted that these sets often fall within what he has termed *separation theory*. All the inequalities of such sets are of the form $x \leqslant y + c$. The residue of a loop whose labeling inequalities are of this form is given by one of $\langle 1, -1, m\rangle$, $\langle -1, 1, m\rangle$, where m is the sum of the constants c around the loop. The graph

for a set S in separation theory is thus its own closure, so the main theorem of the last section reduces, in this case, to Pratt's observation that such a set S is infeasible if and only if the sum of the constants around some simple loop is negative. Pratt notes that this condition can be tested in order $(|V| + |E|)^3$ time by taking a max/+ transitive closure of the incidence matrix of the graph. In practice, however, it may be more efficient to generate loops using one of the algorithms mentioned earlier.

Note that a set of inequalities in separation theory with integer constants is integer feasible if and only if it is real feasible. While the main theorem therefore decides integer feasibility in this case, it cannot decide integer feasibility in general. It has been observed [21], however, that the transformations Bledsoe [3] describes for reducing formulas in integer arithmetic to sets of inequalities tends to produce sets that are integer feasible if and only if they are real feasible. The main theorem thus provides a useful, but not complete, test for integer feasibility.

5. Strict Inequalities

The procedure is trivially generalized to handle strict inequalities (i.e., inequalities of the form $ax + by < c$). Let an admissible loop be *strict* if one or more of its edges is labeled with a strict inequality. A strict loop P with residue $\langle a_p, b_p, c_p \rangle$ is *infeasible* if $a_p + b_p = 0$ and $c_p \leq 0$. If the definition of closure is now modified in such a way that new edges arising from strict loops are labeled with strict inequalities, the main theorem still holds.

6. Extension to Arbitrary Sets of Inequalities

The method can be further generalized to sets of inequalities with arbitrary coefficients and arbitrary numbers of variables.

The basic idea is illustrated by the following example. Consider the set

$$S = \{x \leq y,\ y \leq z,\ z \leq y - x + 1,\ x \geq 2\} \quad .$$

Note that the inequality $z \leq y - x + 1$ has three variables. As shown in Figure 3, we choose two of the three (say z and y) as the endpoints of the edge corresponding to this

inequality in the graph G for S. The term $(-x + 1)$ becomes the "constant" of this inequality. The residue of the only simple loop $(y\ z\ y)$ is given by

$$\langle 1, -1, 0\rangle * \langle 1, -1, -x + 1\rangle$$

and is computed "symbolically" to obtain $\langle 1, -1, -x + 1\rangle$. Note that this loop is infeasible unless $-x + 1 \geq 0$. If the residue inequality $-x + 1 \geq 0$ is now added to the graph, an infeasible simple loop $(v_0\, x\, v_0)$ results, thus making S unsatisfiable.

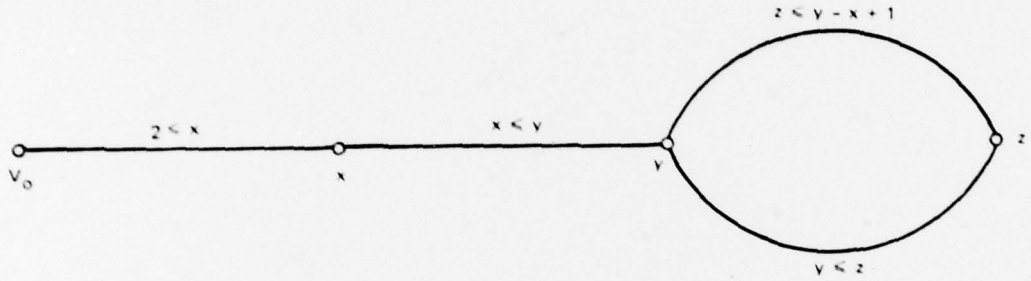


FIGURE 3 GRAPH G FOR $\{x \leq y,\ y \leq z,\ z \leq y - x + 1,\ x \geq 2\}$

We now describe the procedure for an arbitrary set S. We assume that the variables of S other than v_0 are ordered in some way. Each variable that is the lowest or second lowest ranked variable in every inequality in which it appears is said to be a *primary variable*. We adopt the convention that the edge corresponding to a given inequality is always attached to the two nodes corresponding to its primary variables. If it has only one primary variable, one end is attached to v_0, and if it has no primary variables, both ends are attached to v_0. The procedure is as follows:

(1) Compute a closure G' of the graph G for S as usual, evaluating residues "symbolically" as in the example. If G' has an infeasible loop, terminate returning "unsatisfiable." Otherwise, if all the variables of S are primary, terminate returning "satisfiable."

(2) Otherwise, repeat the procedure using the set of residue inequalities of G' in place of S.

Note that the procedure must terminate since the number of non-primary variables must decrease each iteration. One can prove as an extension of the main theorem that the general procedure is complete as well as sound.

R. Tarjan* has observed that any set of inequalities can be polynomially transformed to one with no more than three variables per inequality through the addition of new variables. The inequality $w + x + y + z \leqslant 1$, for example, is replaced by $w + x \leqslant v$, $w + x \geqslant v$, $v + y + z \leqslant 1$. For sets with inequalities having no more than three variables, only two iterations of the procedure are ever required. There does not seem to be any fast way to transform a set of inequalities to one having inequalities with no more than two variables.

7. Proof of the Main Theorem

It follows from Lemma 1 and from the definition of closure that a set S of inequalities (each having no more than two variables) is satisfiable if and only if S′ is unsatisfiable, where S′ labels the edges of a closure of the graph for S. If we define a *closed* graph as one that is a closure of itself, the main theorem can thus be restated as follows:

Theorem: If G is a closed graph for S, then S is satisfiable if and only if G has no infeasible simple loop.

The proof of the theorem requires a number of technical lemmas. Proofs are omitted for the more trivial of these.

Notation: Where P and Q are paths, let PQ denote the concatenation of P with Q.

Lemma 2. If P and Q are admissible paths, then PQ is admissible if and only if b_Q and a_P are of opposite sign.

Notation: Let $T = \langle a, b, c \rangle$ be a triple of reals. Then $T^{\sim}$ denotes the triple $\langle b, a, c \rangle$.

Lemma 3. If T_1, T_2 are triples, $T_1 * T_2 = (\tilde{T}_2 * \tilde{T}_1)^{\sim}$.

Corollary 4. If Q is the reverse of an admissible path P, then $r_P = r_Q^{\sim}$.

Corollary 5. The reverse of an infeasible loop is itself infeasible.

---

*Private Communication.

Lemma 6. Any permutation of an infeasible permutable loop is infeasible.

Pf. Say P is infeasible and P′ is a permutation of P. Then there are paths Q and R such that P = QR and P′ = RQ. Thus,

$$r_P = \langle ka_Q a_R, -kb_Q b_R, k(c_Q a_R - c_R b_Q) \rangle$$

and

$$r_{P'} = \langle k'a_R a_Q, -k'b_R b_Q, k'(c_R a_Q - c_Q b_R) \rangle \quad ,$$

where $k = \frac{a_R}{|a_R|}$ and $k' = \frac{a_Q}{|a_Q|}$. Note that by admissibility of P and P′, both a_R and a_Q are nonzero. By infeasibility of P, $a_R a_Q - b_Q b_R = 0$ and $k(c_Q a_R - c_R b_Q) < 0$.

$$\therefore a_R\left(\frac{c_Q b_R b_Q}{a_Q} - c_R b_Q\right) < 0$$

$$\therefore a_R b_Q\left(\frac{c_Q b_R}{a_Q} - c_R\right) < 0$$

$$\therefore c_R - \frac{c_Q b_Q}{a_Q} < 0 \quad \text{(since } a_R \text{ and } b_Q \text{ have opposite signs)}$$

$$\therefore k'(c_R a_Q - c_Q b_R) < 0 \quad .$$

Recalling that $a_R a_Q - b_Q b_R = 0$, we thus have that P′ is infeasible

Q.E.D.

Notation: Where u, v, w are reals, let $u \overset{w}{<} v$ mean that $u < v$ if $w \geq 0$ and $u > v$ if $w < 0$.

Definition: Where P is an admissible path, the *discriminant* d_P of P is given by $\frac{c_P}{a_P + b_P}$.

Note that an infeasible loop is one with discriminant $-\infty$.

Lemma 7. If PQ is an admissible loop from v_0 to v_0, then PQ is infeasible iff $d_P \overset{a_Q}{\geq} d_Q$ iff $d_P \overset{a_P}{<} d_Q$.

Notation: In the following, let $\langle a_1, b_1, c_1 \rangle$, $\langle a_2, b_2, c_2 \rangle$, $\langle a_3, b_3, c_3 \rangle$, and $\langle a_P, b_P, c_P \rangle$, respectively, denote the residues of P_1, P_2, P_3, and P.

Lemma 8. If G is closed and has an infeasible loop from v_0 to v_0, G has an infeasible simple loop.

Pf. Let P be a shortest infeasible loop from v_0 to v_0 in G. If P is simple, we are done. Otherwise, since, by admissibility, the intermediate vertices of P are distinct from v_0, P can be expressed $P_1 P_2 P_3$, where P_2 is simple. We claim that P_2 is also infeasible.

Suppose not. Then either $a_2 + b_2 = 0$ and $c_2 \geq 0$, or d_{P_2} is finite. In the former case, a_2 and b_2 have opposite signs. It follows from Lemma 2 that b_1 and a_3 must as well, hence $P_1 P_3$ is admissible. Now since

$$r_{P_1 P_2} = \langle 0, b_1, c_1 \rangle * \langle a_2, b_2, c_2 \rangle = \frac{a_2}{|a_2|} \langle 0, -b_1 b_2, c_1 a_2 - c_2 b_1 \rangle \quad ,$$

we have:

$$d_{P_1 P_2} = \frac{c_1 a_2 - c_2 b_1}{-b_1 b_2} = \frac{c_2}{b_2} - \left(\frac{a_2}{b_2}\right) d_{P_1} = \frac{c_2}{b_2} + d_{P_1} \quad .$$

Since P is infeasible, we have from Lemma 7 that

$$\frac{c_2}{b_2} + d_{P_1} \overset{a_3}{\geq} d_{P_3} \quad .$$

Thus,

$$c_2 + b_2 d_{P_1} \overset{a_3 b_2}{>} b_2 d_{P_3}$$

$\therefore c_2 + b_2 d_{P_1} < b_2 d_{P_3}$ (since a_3 and b_2 have opposite signs)

$\therefore b_2 d_{P_1} < b_2 d_{P_3}$ (since $c_2 \geq 0$)

$\therefore d_{P_1} \overset{b_2}{<} d_{P_3}$

$\therefore d_{P_1} \overset{a_3}{>} d_{P_3}$ (since b_2 and a_3 are of opposite sign).

But then $P_1 P_3$ is infeasible by Lemma 2, contradicting our assumption that P is the shortest such loop.

Now if d_{P_2} is finite, the closedness of G provides that some vertex x on P_2 must be connected to v_0 via an edge E labeled $ax \leq c$, where c/a is the discriminant of some cyclic permutation P_2' (possibly = P_2) of P_2. We now have three cases:

Case I. P_2 is not permutable.

Then $P_2' = P_2$, $a = a_2 + b_2$, $c = c_2$, and by Lemma 2, a_2 and b_2 are of the same sign. Also, a must be of this sign; hence both $P_1 E$ and EP_2 are admissible. An argument similar to the one above gives that one or the other of $P_1 E$, EP_2 must be infeasible, contradicting the shortness of P.

Case II. P_2 is permutable and $P_2' = P_2$.

In this case, we have from Lemma 2 that a_2 and b_2 have opposite signs; hence b_1 and a_3 do as well. An argument similar to that given earlier shows that one of $P_1 P_3$, $P_1 E$, and EP_2 must be infeasible, again contradicting the shortness of P_2.

Case III. P_2 is permutable and $P_2' \neq P_2$.

Let P_4 be the initial subpath of P_2 which terminates at x, and let P_5 be the final subpath of P_2 which originates at x (so that $P_2 = P_4 P_5$). In this case, it can be shown that $P_1 P_3$ is admissible, that one of $P_1 P_4 E$, $EP_5 P_3$ is admissible, and that one of these three paths must be infeasible. The shortness of P is thus once again contradicted.

Q.E.D.

Theorem. Let G be a closed graph for S. Then S is satisfiable if and only if G has no simple infeasible loop.

Pf. It follows from Lemma 1 that, if G has a simple, infeasible loop, S must be unsatisfiable. Conversely, suppose G has no such loop. We will show that S is satisfiable by constructing a solution.

Let $v_1, \ldots, v_r$ be the variables of S other than v_0. We construct a sequence $\hat{v}_0, \hat{v}_1, \ldots, \hat{v}_r$ of reals and a sequence $G_0, G_1, \ldots, G_r$ of graphs inductively as follows:

(1) Let $\hat{v}_0 = 0$ and $G_0 = G$.

(2) Suppose $\hat{v}_i$ and G_i have been determined for $0 \leq i < j \leq r$. Let $sup_j = \min\{d_P | P \text{ is an admissible path from } v_j \text{ to } v_0 \text{ in } G_{j-1} \text{ and } a_P > 0\}$. $inf_j = \max\{d_P | P \text{ is an admissible path from } v_0 \text{ to } v_j \text{ in } G_{j-1} \text{ and } b_P < 0\}$. (where it is understood that $\min\emptyset = \infty$ and $\max\emptyset = -\infty$). Then let $\hat{v}_j$ be any value in the interval $[inf_j, sup_j]$. (We show momentarily that $inf_j \leq sup_j$.) Let G_j be obtained from G_{j-1} by adding two new edges from v_j to v_0, labeled $v_j \leq \hat{v}_j$ and $v_j \geq \hat{v}_j$, respectively.

To ensure that the $\hat{v}_j$'s and G_j's are well defined, we must show that, for $1 \leq j \leq r$, $inf_j \leq sup_j$. It will then remain to show that the $\hat{v}_j$'s do indeed give a solution for S.

We need the following claim:

Claim. (i) For $1 \leq j \leq r$, $inf_j \leq sup_j$

(ii) For $0 \leq j \leq r$, G_j has no infeasible simple loops.

Pf. By induction on j.

Basis. $j = 0$.

In this case, (i) holds vacuously, and (ii) holds since $G_0 = G$.

Induction Step. $0 < j \leq r$.

For (i), suppose, to the contrary, that $inf_j > sup_j$. Then in G_{j-1} admissible paths P_1, P_2 exist from v_0 to v_j and v_j to v_0, respectively, with $b_{P_1} < 0$,

$a_{P_2} > 0$, and $d_{P_1} > d_{P_2}$. By Lemma 2, P_1P_2 is an admissible loop, and by Lemma 7, P_1P_2 is infeasible. By Lemma 8, then, G_{j-1} has a simple infeasible loop, contradicting (ii) of the induction hypothesis.

For (ii), suppose G_j has an infeasible simple loop P. Since G_{j-1} has no such loop, and since the loop formed by the two new edges added to G_{j-1} to obtain G_j is not infeasible, P (or its reverse) must be of the form P'E, where E is one of the two new edges (say the one labeled $v_j \leqslant \hat{v}_j$; the other case is handled similarly), and P' is a path from v_0 to v_j in G_{j-1}. But then, by Lemma 7, $d_{P'} > d_E = \hat{v}_j$, contradicting $\hat{v}_j \geqslant \inf_j \geqslant d_{P'}$. (Note that $b_{P'} < 0$ from the admissibility of P'E.)

Q.E.D.

It now remains to show that the $\hat{v}_j$'s satisfy S. So let $ax + by \leqslant c$ be an inequality of S. We claim that $a\hat{x} + b\hat{y} \leqslant c$. We treat the case in which $a > 0$ and $b < 0$; the other cases are argued similarly. Let E be the edge labeled $ax + by \leqslant c$ in G_r. Then, where E_1 is the edge labeled $\hat{x} \leqslant x$ in G_r, and E_2 is the one labeled $y \leqslant \hat{y}$, E_1EE_2 forms an admissible loop. The residue of this loop is

$$\langle 0, -1, -\hat{x}\rangle * \langle a, b, c\rangle * \langle 1, 0, \hat{y}\rangle = \langle 0, 0, -a\hat{x} - b\hat{y} + c\rangle \quad .$$

Since, by the claim proved above, and by Lemma 8, G_r has no infeasible loops from v_0 to v_0, we have $-a\hat{x} - b\hat{y} + c \geqslant 0$. Thus $a\hat{x} + b\hat{y} \leqslant c$ as required.

Q.E.D.

Acknowledgments

The author gratefully acknowledges the insights provided by R. Tarjan, R. Boyer, J. Moore, and M. W. Green.

REFERENCES

1. W. W. Bledsoe. Program correctness. The University of Texas at Austin Mathematics Department Memo ATP-14 (January 1974).

2. W. W. Bledsoe. The sup-inf method in Presburger arithmetic. The University of Texas at Austin Mathematics Department Memo ATP-18 (December 1974).

3. W. W. Bledsoe. A new method for proving certain Presburger formulas. Advance Papers, 4th Int. Joint Conf. on Artif. Intell., 15-21, Tibilisi, Georgia U.S.S.R (September 1975).

4. W. W. Bledsoe, R. S. Boyer, and W. H. Henneman. Computer proofs of limit theorems. *Artif. Intell.* **3**, 27-60 (1972).

5. W. W. Bledsoe and P. Bruell. A man-machine theorem-proving system. *Artif. Intell.* **5**, 51-72 (1974).

6. D. C. Cooper. Programs for mechanical program verification. In *Mach. Intell.* **6**, 43-59, American Elsevier, New York (1971).

7. G. B. Dantzig. *Linear Programming and Extensions*, Princeton University Press, Princeton, New Jersey (1962).

8. L. P. Deutch. An interactive program verifier. Ph.D. Thesis, University of California, Berkeley (1973).

9. B. Elspas, R. E. Boyer, R. Shostak, and J. Spitzen. A verification system for JOVIAL/J3 programs. SRI Technical Report 3756-1, Stanford Research Institute, Menlo Park, California (January 1976).

10. R. E. Gomory. An algorithm for integer solutions to linear programs. Princeton IBM Math. Res. Report (November 1958); also in R. L. Graves and P. Wolfe (eds.), *Recent Advances in Mathematical Programming*, 269-302, McGraw-Hill, New York (1963).

11. D. I. Good, R. L. London, and W. W. Bledsoe. An interactive verification system. Proc. 1975 Int. Conf. on Reliable Software, Los Angeles (April 1975).

12. S. Igarashi, R. L. London, and D. C. Luckham. Automatic program verification 1: A logical basis and its implementation. Stanford AI Memo 200 (May 1973) and USC Information Sciences Institute Rept. ISI/RR-73-11 (May 1973).

13. D. B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM J. Computing* **4**, 77–84 (1975).

14. J. King. A program verifier. Ph.D. thesis, Carnegie-Mellon University, Pittsburgh (1969).

15. R. D. Lee. An application of mathematical logic to the integer linear programming problem. *Notre Dame J. Formal Logic* **XIII**, 2 (April 1972).

16. S. D. Litvintchouk and V. R. Pratt. A proof checker for dynamic logic. 5th Int. Joint Conference on Art. Int., 552–558, Boston (August 1977).

17. M. Prabhaker and N. Deo. On algorithms for enumerating all circuits of a graph. *SIAM J. Computing* **5**, 1 (March 1976).

18. V. R. Pratt. Two easy theories whose combination is hard. M.I.T. Technical Rept., Cambridge, Massachusetts (September 1977).

19. R. C. Read and R. E. Tarjan. Bounds on backtrack algorithms for listing cycles, paths, and spanning trees. ERL Memo M 433, Electronics Research Laboratory, University of California, Berkeley (1973).

20. R. Shostak. An efficient decision procedure for arithmetic with function symbols. Presented at 5th Int. Joint Conf. on Art. Int., Cambridge, Massachusetts (August 1977).

21. R. Shostak. On the sup-inf method for proving Presburger formulas. *J. ACM* **24**, 4, 529–543 (October 1977).

22. N. Suzuki. Verifying programs by algebraic and logical reduction. *Proc. Int. Conf. on Reliable Software* (Sigplan Notices) **10**, 6 (June 1975).

23. J. L. Szwarcfiter and P. E. Lauer. Finding the elementary cycles of a directed graph in O(n + m) per cycle, No. 60, University of Newcastle upon Tyne, Newcastle upon Tyne, England (May 1974).

24. R. Tarjan. Enumeration of the elementary circuits of a directed graph. *SIAM J. Computing* **2** (1973).

25. R. J. Waldinger and K. N. Levitt. Reasoning about Programs. *J. Artif. Int.* **5**, 235–316 (1974).